Conservation Functions for 1-D Automata: Efficient Algorithms, New Results, and a Partial Taxonomy

(Abbreviated title: "New Conservation Laws for 1-D Automata")

Leemon Baird (leemon.baird@usafa.edu)
Barry Fagin (barry.fagin@usafa.edu)
Department of Computer Science
US Air Force Academy
Colorado Springs, CO 80840
719-333-3590

Abstract:* We present theorems that can be used for improved efficiency in the calculation of conservation functions for cellular automata. We report results obtained from implementations of algorithms based on these theorems that show conservation laws for 1-D cellular automata of higher order than any previously known. We introduce the notion of trivial and core conservation functions to distinguish truly new conservation functions from simple extensions of lower-order ones. We then present the complete list of conservation functions up to order 16 for the 256 elementary 1-d binary cellular automata. These include CAs that were not previously known to have nontrivial conservation functions.

Keywords: Cellular automata, conservation functions, linear algebra, classification scheme, taxonomy

I. INTRODUCTION

Conservation functions indicate fundamental physical properties of systems, and are therefore eagerly sought after in science. Conservation of energy and conservation of angular momentum, for example, are two fundamental principles at different physical levels that offer profound insights into the nature of the world we live in.

Similarly, conservation functions of dynamic systems, when they can be found, offer insights into the evolution of the system over time. The simplest conservation functions indicate rules on the number of organisms, but more complex ones such as conservation of "energy", where energy is suitably defined, suggest something about the inherent nature of the system under study. Cellular automata with identical conservation functions are, in some sense, closely related to one another. Thus conservation functions can be used to classify cellular automata, and to identify connections between seemingly unrelated systems.

The classification scheme we present here permits finer distinctions between CAs than merely their lowest order of conservation function (as has been done in the past) by grouping functions together with identical sets of *basis vectors* and by distinguishing *core* functions from *simple extensions*. Of the 88 equivalent 1-dimensional cellular automata, 8 have provably trivial conservation functions only, 47 have conservation functions of various kinds, and 33 remain unclassified. Those that we have classified include CAs with newly discovered conservation functions of order 8, 9, 12, 13 and 14. Our calculations also show there are no CAs with conservation functions of order 15 or 16. We can therefore present here the complete classification scheme for 1-dimensional cellular automata based on conservation functions up through order 16.

II. TERMINOLOGY

A one-dimensional *cellular automaton* is an array of finite size W of cells containing {0,1…*s*-1) . The array is called the *universe,* and the number of cells it contains is the size of the universe. The universe is considered to have no boundaries, which means in one dimension the end cells are considered adjacent to one another.

The current set of values in each cell is the *state* of the universe. This state changes over time based on a *characteristic function*, parameterized by a neighborhood size *n.* To determine the next state of any cell, the state of the (*n*-1)/2 cells on each side of the current cell, along with the state of the current cell itself is examined, and rules applied that uniquely determine the new state. (Since we are only interested in symmetric neighborhoods, n will always be odd). The new state of the universe as determined by the parallel application of the characteristic function is called the *successor state*. A 1-dimensional CA with *s* states and neighborhood of size *n* can be configured in ${}_{s}s^{n}$ ways, not all of which are unique.

Let $\mathbf{x}$ be the state of the universe at a given time, and $\mathbf{x}_i$ be the state of cell *i*. An energy function $E(\mathbf{x})$ is defined as:

$$E(\mathbf{x}) = \sum_{i=0}^{n-1} f\left(\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots \mathbf{x}_{i+m-1}\right)$$

where subscripts outside the universe wrap around. The energy function is a sum of the energy of *n* different regions, each of which contains *m* cells. We refer to *m* as the *order* of the function. Let succ(**x**) be the successor state of the universe reached on the next time step if it starts in state **x**. We say that *E* is conserved, or that *E* is an energy conservation function, iff, for all finite universes:

Conserved: $\forall x \quad E(\mathbf{x}) = E(succ(\mathbf{x}))$

This is equivalent to the following [1]:

$$\forall \mathbf{x}, \mathbf{y} \quad E(\mathbf{x}) - E(succ(\mathbf{x})) = E(\mathbf{y}) - E(succ(\mathbf{y}))$$

where **x** and **y** are states that differ by exactly one cell.

It can be shown that for neighborhood size of *n=3,* we need only analyze the energy of a *2m+3*-cell array and the *2m+1*-cell array of its successor state in order to determine if a conservation law exists, despite the fact that the universe itself may be arbitrarily large. This is what makes the computation of conservation functions possible.

It is a theorem of Hattori and Takesue [1] that if we zero the block of cells in **x** and **y** either to the left or the right of the non-identical cell, our energy calculations will be unaffected. Thus applying this theorem over all possible *(m+1)*-cell states of the universe for a given CA gives a system of $s^{(m+1)}$ equations in s^m unknowns, with the right hand side equal to zero. If this system of equations has a solution (that is, if the resulting matrix has a non-empty null space), then an energy conservation function of order *m* has been found. If it does not, then no such function exists for that CA.

Conservation laws are fundamental properties eagerly sought in studies of physical phenomena. Conservation functions are equally important for mathematical objects because they say something fundamental about the entity under study. Cellular automata can be classified by the order of their conservation functions, and can be understood more deeply by the nature of the quantities conserved. CAs are simple objects that generate surprisingly complex behavior. Conservation functions can shed light on how such behavior might emerge.

III. DEFINITIONS AND THEOREMS

Clearly energy functions that assign the same value to all states are conserved. We call such functions *trivial.* Of greater interest are non-trivial energy conservation functions. It is useful in the calculation of conservation functions to eliminate trivial functions from the solution space. To do this, we will need the theorems that follow.

We will view the rows of a matrix as vectors, and the matrix as a set of vectors that spans some space. It is useful to first define the function *span*() that gives that space.

Definition 1: For any *n*-column matrix **T,** *span*(**T**) is the set of vectors in $\mathbf{R}^n$ that are linear combinations of the rows of **T**.

Let *null*() be a function that maps a basis for one subspace to a basis for its null space. We will assume that the bases are represented as matrices, whose rows are vectors in those spaces.

Definition 2: A valid *null* function is any function from matrices to matrices that has the following three properties for all *n*-column matrices **T**:

$$span(\mathbf{T}) \cap span(null(\mathbf{T})) = \{\mathbf{0}\}$$

$$span\left(\begin{bmatrix} \mathbf{T} \\ comp(\mathbf{T}) \end{bmatrix}\right) = \Re^n$$

each row of *null*(**T**) is orthogonal to each row of **T**

This definition does not require that the rows of *null*(**T**) be unit length, orthogonal to each other, or even linearly independent. Given this definition, *span*(*null*(**T**)) is uniquely defined (it's the null space of *span*(**T**)), but *null*(**T**) itself is not uniquely defined. Note that in the above, **0** is the all-zeros vector, which will be an element of *span*(**T**) for any **T**.

Many different *null* functions can satisfy this definition. Three different examples of such functions are the Mathematica function `NullSpace[T]`, the MATLAB function `null(T)`, and the MATLAB function `null(T, 'r')`. These functions each return a different basis set for the null space of **T**. In general, if the elements of each row of the matrix **T** are considered to be the coefficients of a linear equation equal to zero (a homogenous linear equation), then **T** is a set of simultaneous equations, each row of *null*(**T**) gives one possible solution to that set of equations, and all possible solutions can be found by taking linear combinations of the rows of *null*(**T**).

It is also useful to consider *complement* functions, which are defined like the *null* functions, except without the third constraint.

Definition 3: A valid *complement* function is any function from matrices to matrices that has the following two properties for all *n*-column matrices **T**:

$$span(\mathbf{T}) \cap span(comp(\mathbf{T})) = \{\mathbf{0}\}$$

$$span\left(\begin{bmatrix} \mathbf{T} \\ comp(\mathbf{T}) \end{bmatrix}\right) = \Re^n$$

This is identical to the definition of the *null* function, except without the orthogonality constraint. Although *span*(*null*(**T**)) is uniquely defined, *span*(*comp*(**T**)) is not. For example, in the plane, if **T** spans a line through the origin, then *null*(**T**) spans the orthogonal line through the origin, and *comp*(**T**) spans some line through the origin that is non-collinear with *span*(**T**), but not necessarily orthogonal to it.

The use of complementary matrices can greatly simplify the sets of equations for conservation functions. Suppose the rows of a matrix **C** are thought of as coefficients of equations defining conserved energy functions, and the rows of a matrix **T** define energy functions that are trivial. Then *span*(*null*(**C**)) is the space of all conserved functions, and *span*(*null*(**T**)) is the space of all trivial energy functions.

We are interested only in the nontrivial conserved functions. It would be nice to have a matrix **N** that spanned exactly the nontrivial conserved functions. Unfortunately, that is not possible, because the nontrivial conserved functions do not form a subspace. Instead, we will construct a matrix **N** that spans a subset of the conserved functions, that does not include any trivials (other than the origin), and that together with **T** spans all of the conserved functions. If the space of trivials is known, then the conserved nontrivials for a given CA can be defined completely by simply giving such an **N**.

One obvious way to choose such an N is

$$\mathbf{N} = null\left(\begin{bmatrix} \mathbf{C} \\ null(\mathbf{T}) \end{bmatrix}\right)$$

The top half of the matrix is **C**, which contains equations requiring the functions to be conserved. The bottom half is *null*(**T**). If it were just **T**, it would require the functions to be trivial. Since it is *null*(**T**), it requires the functions to be nontrivial. The outer *null*() finds a basis for the space of all solutions to those equations, which will be both conserved and nontrivial. So *span*(**N**) will be a space of nontrivial, conserved functions, though it will not include all such functions. Together, **N** and **T** will span the entire space of conserved functions.

Note that calculating **N** requires more computation than just calculating null(**C**). It is also more difficult to analyze. Therefore, we seek an alternative **N** that can be calculated more simply. One obvious choice might be to replace null(**T**) with comp(**T**) in the definition of **N**. The following theorem proves that this gives all the desired properties.

Theorem 1: If *null*() and *comp*() are valid null and complement functions respectively, and **T** and **C** are *n*-column matrices, and *span*(*null*(**T**)) $\subseteq$ *span*(*null*(**C**)), and **N** is defined as:

$$\mathbf{N} = null\left(\begin{bmatrix} \mathbf{C} \\ comp(\mathbf{T}) \end{bmatrix}\right)$$

then both of the following hold:

$$span(\mathbf{N}) \cap span(null(\mathbf{T})) = \{\mathbf{0}\}$$

$$span\left(\begin{bmatrix} \mathbf{N} \\ null(\mathbf{T}) \end{bmatrix}\right) = span(null(\mathbf{C}))$$

Proof: The first property is proved as

$$span(\mathbf{N}) \cap span(null(\mathbf{T}))$$

$$= \ span\left(null\left(\begin{bmatrix} \mathbf{C} \\ comp(\mathbf{T}) \end{bmatrix}\right)\right) \ \cap \ span(null(\mathbf{T}))$$

$$= (\ span(null(\boldsymbol{C})) \cap span(null(comp(\boldsymbol{T})))\ ) \cap span(null(\boldsymbol{T}))$$

$$= span(null(\boldsymbol{C})) \cap (\ span(null(comp(\boldsymbol{T})) \cap span(null(\boldsymbol{T}))\ )$$

$$= span(null(\boldsymbol{C})) \cap \{\mathbf{0}\}$$

$$= \{\mathbf{0}\}$$

To prove the second property, we show that the left side is a subset of the right side, then show that the right side is a subset of the left.

That the left is a subset of the right is obvious. Any element of *span*(*null*(**T**)) must be an element of *span*(*null*(**C**)) by assumption. Any element of *span*(**N**) must be an element of *span*(*null*(**C**)) because of the **C** in the top half of the matrix in the definition of **N**. Therefore the left side is a subset of the right.

To prove the right side is a subset of the left, consider an arbitrary vector $\mathbf{v} \in \mathit{span}(\mathit{null}(\mathbf{C}))$. Because $\mathit{null}(\mathbf{T})$ and $\mathit{null}(\mathit{comp}(\mathbf{T}))$ together span $\mathbf{R}^n$, it is possible to decompose $\mathbf{v}$ as

$$\mathbf{v} = \mathbf{v}_t + \mathbf{v}_c$$

where

$$\mathbf{v}_t \in \mathit{span}(\mathit{null}(\mathbf{T}))$$
$$\mathbf{v}_c \in \mathit{span}(\mathit{null}(\mathit{comp}(\mathbf{T})))$$

Note that $\mathbf{v}_c \in \mathit{span}(\mathbf{N})$ because of the bottom half of the definition of $\mathbf{N}$. $\mathbf{v}_t \in \text{span}(\text{null}(\mathbf{T}))$ by definition. Therefore:

$$\mathbf{v} \in \mathit{span}\left(\begin{bmatrix} \mathbf{N} \\ \mathit{null}(\mathbf{T}) \end{bmatrix}\right)$$

because it is a sum of one vector from the top space and one from the bottom. So the left side is a subset of the right, and therefore the second equality holds, and so the entire theorem is proved. □

We now need a *comp*() function that will simplify the definition of **N**. For any given **T** matrix, it will sometimes be possible to find a *comp*(**T**) matrix where every row consists of all zero elements except for a single 1. In that case, the following lemma allows us to greatly simplify **N**.

Lemma 1: If **A** and **B** are *n*-column matrices, and each row of **B** consists of a single 1 element and the rest zeroes, and **S** is a set of the indices of the columns in **B** that have 1 elements in them, and $\mathbf{A}_s$ is the **A** matrix with all the columns in **S** removed, and a superset **S** as in $\mathbf{X}^S$ represents the set of vectors **X** with zeros inserted to make them *n*-element vectors with zeros in the **S** positions, then:

$$\mathit{span}(\mathit{null}\left(\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}\right) = \mathit{span}(\mathit{null}(\mathbf{A}_s))^S$$

Proof: Each row of **B** can be considered an equation that states that one particular variable is equal to zero. So when **A** and **B** are concatenated, **B** forces certain variables to zero, which is equivalent to deleting columns of **A** before the *null*() operation, then adding zeros back in afterwards. □

If a *comp*(**T**) matrix can be found that has a single 1 per row, then this lemma allows us to calculate **N** even more easily than calculating *null*(**C**). Rather than calculate the nullspace of **C**, we can calculate the nullspace of the matrix formed by deleting certain columns of **C**. In fact, for 1D CAs, this eliminates half the columns of **C**, making the null calculation much faster. So we now turn to the question of what *comp*(**T**) matrices can be formed that have a single 1 per row.

It turns out that for 1D CAs there is a very useful choice for *comp*(**T**). It will delete the entire left half of the **C** matrix. That corresponds to imposing the following constraint on the energy function f:

$$\forall \mathbf{X} \in \{0,1\}^{n-1} \quad f(0\mathbf{X}) = 0$$

Theorem 1 implies that, for a 1D, 2-state, neighborhood 3 cellular automaton, the following set of 2^{m-1} trivial energy functions form a *basis set* for all possible trivial energy functions over m bits, where $1<S<2^{m-1}$. and the expression 0S (1S) represents a 0 (1) followed by the bits of S in binary:

$$f_0(x) = 1$$

$$f_{2^{m-1}}(x) = \begin{cases} 1 \text{ if } x = 000\ldots0001 \\ -1 \text{ if } x = 1000\ldots000 \\ 0 \text{ otherwise} \end{cases}$$

$$f_S(x) = \begin{cases} 1 \text{ if } x = 0S \text{ or } x=1S \\ -1 \text{ if } x = S0 \text{ or } x = S1 \\ 0 \text{ otherwise} \end{cases}$$

Theorem 2 implies that for a 1D, 2-state, neighborhood 3 cellular automaton, a basis set for the set of all conserved linear functions over *m* bits can consist solely of energy functions satisfying the constraint f(0S)=0 for all m-1 element strings S. In other words, the energy of an *m*-cell region can be defined to be zero for any region whose leftmost cell is zero.

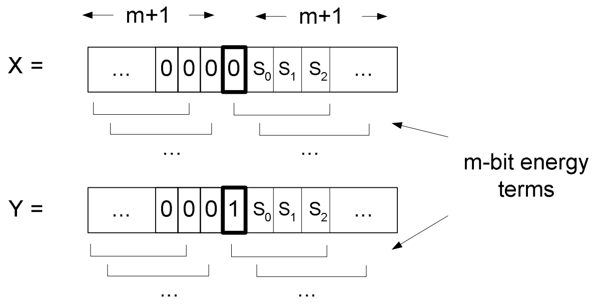
Once a non-trivial conservation function of order *m* has been found, it can be extended in a very simple way to produce a function of order *m+1* by ignoring the newly added cell on the right side. The conservation functions reported in [1], however, do not distinguish between new functions derived in this way, since a conservation function of order *m* is by definition a conservation function of order *n* >= *m* for all n. It seems useful, however, to distinguish between conservation functions derived from existing ones by simply ignoring that newly-added cell, and conservation functions that have a completely different structure from lower-order ones.

We refer to a non-trivial conservation function for a given CA that cannot be derived from non-trivial conservation of lower order in that way as a *core* function. The core function of lowest order for a given CA is the *primary core* function for that CA. We call functions derived from a core by ignoring the new cell *simple extensions.* To eliminate simple extensions from the solution space, we append to the bottom of C the extensions of the cores found for lower orders. This forces the solutions to be orthogonal to those extensions. The resulting solution space is therefore much smaller, but, when combined with all the extensions, the two together span the space of all nontrivial conserved functions.

IV. AN ALGORITHM FOR CALCULATING CONSERVATION FUNCTIONS OF ORDER *M* FOR A GIVEN CA

By examining in detail how each of the energy terms are formed, it is possible to generate the state space matrix without doing explicit lookups and generation of all states and next state vectors. We do this by breaking the state space matrix up into three distinct matrices of different sizes, combining them only at the end when the null space must be calculated.

To see how this is done, consider first the terms in E(X) –E(Y), the energy terms in the X and Y initial states, and what happens to them as the order *m* increases. We have for any *m:*



where S0, S1 … are state counting bits (we have the leftmost bit change the fastest, for reasons that will become clear shortly). The energy expressions E(X) and E(Y) are just the sums of the m-bit energy terms for the X and Y states respectively. It is clear by inspection that all terms in E(X)-E(Y) cancel except for those involving the center bit. Because we may also assume that f(0,…) = 0 for our energy function f, the matrix for E(X)-E(Y) becomes

$–f(1\ S_0\ S_1\ \ldots\ S_{m-2})$

where the S bits are as previously described. In other words, each row has a -1 in the column indicated by bits $1S_0S_1\ldots S_{m-2}$. Making the leftmost 1 implicit in the column numbering, and reversing the column subscripts, we have

For m=1, E(X) –E(Y) = -1

For m=2, E(X) – E(Y) =

S_0	10	11
0	-1	0
1	0	-1

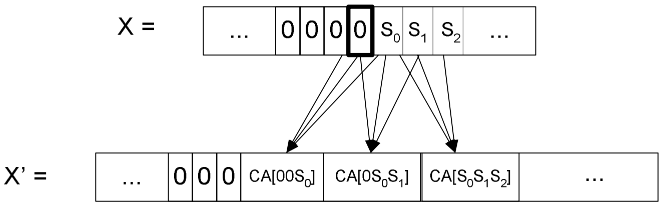
For m=3, E(X) – E(Y) =

S_0S_1	100	110	101	111
00	-1	0	0	0
10	0	-1	0	0
01	0	0	-1	0
11	0	0	0	-1

and in general E(X)-E(Y) is given by a negative identity matrix of 2^{m-1} rows and 2^{m-1} columns.

Now consider the energy terms of the successor states, E(X') and E(Y'). It is useful to first consider just those CAs that map the neighborhood region 000 to the 0 state. We shall refer to these CAs as *zero-preserving* automata.

For zero-preserving automata, we have the following relationship between X and its successor state X':



since zero-preserving automata map 000 to 0. (Here CA[xyz] denotes the appropriate bit of the number of the CA written in binary). A similar analysis holds for states Y and Y'.

For m = 1, we can compute the energy difference of the successor states E(X') – E(Y') as

$\{E(X') - E(Y')\}_1 = CA[00S_0] + CA[0S_0S_1] - CA[001] - CA[01S_0] - CA[1S_0S_1]$

since all other terms cancel, and we assume the energy function f maps all terms with an MSB of 0 to 0.

Let us rewrite the conservation law as

$E(X) - E(Y) + E(Y') - E(X') = 0$

Plugging our formula for m=1 above gives

$-I + CA[001] + CA[01S_0] + CA[1S_0S_1] - CA[00S_0] - CA[0S_0S_1] = 0$

where I is the identity matrix. Ignoring signs, we will refer to these terms as Ex-Ey, D1, D2, D3, R1 and R2.

We reorder this expression to group terms with identical numbers of state bits together, giving

$-I + CA[001] + CA[01S_0] - CA[00S_0] + CA[1S_0S_1] - CA[0S_0S_1] = 0$

or equivalently

Ex-Ey + D1 + D2 – R1 + D3 – R2 = 0

These six terms, with some bookkeeping, can be represented with three matrices of different sizes that grow in similar ways as m increases. Their size differences do not matter until the calculation of a null space is required, at which time they can all be made of uniform size and summed.

To see how this process works, first note that as m increases the energy terms grow to the right. Using the convention of having the leftmost state bit increase first, if we read the CA bit index from right to left, then each increment of m corresponds to simply copying the existing matrix both above itself and horizontally, and then shifting the individual rows of the new matrix by exactly half the columns if a particular bit of the CA is 1.

For example, consider the D1 term for CA = 50 = 00110010 in binary. Recall that we read CA bit indices from right to left. For m=1, we have

$$D1_1 = CA[001] = 1$$

For m = 2, we have

$$D1_2 = CA[001]\ CA[01S_0] \quad = \begin{array}{l} CA[001]\ CA[010] = 1\ 0 \\ CA[001]\ CA[011] = 1\ 0 \end{array}$$

$$= \begin{array}{c|cc} S_0 & 10 & 11 \\ \hline 0 & 1 & 0 \\ 1 & 1 & 0 \end{array}$$

meaning for the state with $S_0 = 0$ this matrix has an energy term of f(10), just as it does for the state with $S_0 = 1$ (recalling again that the MSB of 1 in the energy term is implicit in the column numbering). Note that we obtained $D1_2$ from D1 by copying the rows, doubling the columns, and moving the individual columns over to the right half of the matrix based on the values of the vector $CA[01S_0]$ = (CA[010] CA[011]} = {0 0}. We refer to this doubling and shifting operation as *expansion via the vector* {0 0}, and refer to {0 0} as the *expansion vector*. In this case both values in the expansion vector are zero so no shifting of the columns occurs.

For m = 3, we have

$D1_3 = CA[001]\ CA[01S_0]CA[1S_0S1]$

$$
= \begin{array}{l}
CA[001]\ CA[010]\ CA[100] \\
CA[001]\ CA[011]\ CA[110] \\
CA[001]\ CA[010]\ CA[101] \\
CA[001]\ CA[011]\ CA[111]
\end{array}
= \begin{array}{c}
101 \\
100 \\
101 \\
100
\end{array}
$$

$$
= \begin{array}{c|cccc}
S_0S_1 & 100 & 110 & 101 & 111 \\
\hline
00 & 0 & 0 & 1 & 0 \\
10 & 1 & 0 & 0 & 0 \\
01 & 0 & 0 & 1 & 0 \\
11 & 1 & 0 & 0 & 0
\end{array}
$$

meaning for the state with $S_0S_1 = 00$ this matrix contributes an energy term of f(101), and so forth. This time the expansion vector is $CA[1S_0S_1] = \{CA[100]\ CA[110]\ CA[101]\ CA[111]\} = \{1\ 0\ 1\ 0\}$, corresponding to the third non-zero block in the previous figure.

After m = 3, the expansion vector always $CA[S_{m-4}S_{m-3}S_{m-2}]$. So D1 is now completely determined by:

$D1_1 = CA[001]$
$D1_2 = expansion\_via(D1_1, CA[01S_0])$
$D1_3 = expansion\_via(D1_2, CA[1S_0S_1])$
$D1_m = expansion\_via(D1_{m-1}, CA[S_{m-4}S_{m-3}S_{m-2}])$ for m > 3

It can be shown that after m=1, D3 and R2 have identical expansion vectors and may therefore be combined into a single matrix C = D3-R2. Similarly, after m=2, D2 and R1 may be combined into a single matrix B = D2 - R1. Ex-Ey may be combined immediately with D1 into A = Ex-Ey+D1. Thus after some preliminary bookkeeping up through m=3, we need only maintain three energy term matrices A, B and C using only expansion operations with vectors derived from the CA under test. When the desired *m* has been reached, we replicate A 4 times and B twice, and then solve the $2^{m+1} \times 2^{m-1}$ system of equations A+B+C = 0.

We refer to the matrix A+B+C as the *state space matrix N.* Finding a conservation function of order m for a given CA requires calculating the null space of N.

We see from the method of construction that A, B and C are sparse: Each one is the sum of matrices with only one non-zero term per row. This means that N can have at most six non-zero terms per row, out of rows that contain 2^{m-1} entries. In practice, N also contains a large number of redundant rows. Empirical analysis shows that removing all duplicate rows from N reduces its size by about a factor of two.

The authors have implemented the calculation of the null space of N for a given CA and order *m* in MATLAB. This program has been run at various DoD High Performance Research Center sites. We now turn to a description of the results we have obtained.

V. A TAXONOMY FOR 1-D CELLULAR AUTOMATA BASED ON CONSERVATION LAWS

We now present a complete taxonomy of 1-d CAs based on conservation functions of order m <= 16. We use Wolfram's numbering scheme [3] to denote particular CAs, in which the successive state *b* of a cell appears as bit number *i* in an 8-bit binary number that designates the CA, where *i* is itself a 3-bit binary number determined by the left, center and right cell contents in a CA's next state rules.

When examining all *s*-state CAs*,* symmetry laws can be seen to divide the set into equivalence classes. Viewing the CA through a mirror produces a CA with identical properties, as does replacing state number *j* with state symbol *s-j.* For binary CAs, repeated application of these laws divides the 256 automata into 88 equivalence classes. For purposes of this discussion, if a single CA is mentioned, it is understood to refer to all CAs in its class. We will normally use the lowest numbered CA of a class to represent it.

Some CAs can be shown to have no non-trivial energy conservation functions. In particular, it can be shown that all CAs with next state functions of x0x0x000 have only trivial energy conservation functions, where x denotes 0 or 1.

A. Classification By Lowest Order Conservation Function

We first present a table of CAs grouped by their lowest order conservation function. Table 1 lists all CAs with conservation functions up through order 16, grouped by the order at which their first conservation function emerges. With one exception, this table is identical at lower orders to that in [1], but extends it to significantly higher values of *m.*

We note that [1] reports the existence of a CA with a conservation function at m=7. Wolfram [3] disagrees, and our results support his conclusions. We believe the entry in Table 1 in [1] for CA 19h and its equivalence class at m=7 should be 0.

There are 3 CA equivalence classes with conservation functions of order 1: The shifter (170), the identity (204), and 184. The CA 184 is discrete asymmetric exclusion, and has been extensively studied [4].

At m=2 there are 11 equivalence classes with second order conservation properties, 8 at m=3, and so forth. The results for m >= 8 are, to our knowledge, new ([5] reports higher order results for reversible CAs only). In addition to the new results, the use of basis vectors and the resulting taxonomy that emerges distinguishes our work from [6], [7], [8] and [9].

B. Classification By Conservation Function Basis

We can gain more insight into the structure of CA conservation functions by examining the basis of the associated null space, noting both the number of dimensions and the basis vectors themselves. The tables in this section show the number of dimensions in the null space (which is equivalent to the dimensionality of the basis of the conservation function), and groups the CAs together for which the basis functions are identical. Such CAs should be regarded as equivalent in some natural sense. We can also examine what non-simple core functions emerge after the first one, as *m* increases.

Table 2 shows the non-zero half of the conservation function basis vectors for CAs with primary cores at m=2 through 5. (The three CAs with primary cores at m=1 conserve all functions on m bits and are therefore ignored). Entries that appear to the right of the first non-empty column represent functions that are also conserved and are not simple extensions of any cores to the left. For example, the CA of 15 conserves the m=2-bit function f([10 11]) = [1 0], meaning that it conserves all “10” bit pairs. It also conserves the 3-bit function f([100 101 110 111]) = [-1 1 -2 0], and so forth. CAs with identical basis vectors for a given m are grouped by color. This table and those that follow have the property that all m-bit conservation functions for the indicated CA are some linear combination of trivial functions and the basis vectors shown.

The second-order conservation functions are exactly those identified in [2]. We believe all save 200 belong together since they have identical basis vectors. Fuks notes that all CAs with second-order conservation functions save 200 have identical invariants. Our results are equivalent.

Tables 3 and 4 show the dimensionality of null spaces and equivalence classes of CAs for all cores currently identified by our code.

VI. CONCLUSION

It is surprising that, given how well-studied these simple automata are, new conservation functions remain to be discovered. A simple taxonomy for them, analogous to taxonomies for living systems, has yet to be found.

Classifying existing systems by the basis vectors of their conservations may point the way to a more exhaustive taxonomy. We would argue that CAs with identical basis vectors for a conservation function of a given order are related to one another in some important sense.

A total of 33 equivalence classes of the 1-D binary cellular automata remain to be classified. For these CAs, neither impossibility proofs nor existence proofs for conservation functions are currently available. Our classification scheme is thus incomplete; we have only determined by exhaustive search that no conservations of order 16 or below exist beyond those described.

The existence of an upper bound on the order of conservation functions for one-dimensional CAs of a given neighborhood size remains an open problem. If the state space matrix constructed as described has full rank, then it has no null space and therefore the associated CA has no conservation function. However, a proof that does not require explicit construction and analysis of the matrix remains elusive.

The computational techniques described here may also be applied to higher dimensional binary automata, such as Conway's Game of Life, to determine the presence of conservation functions. At present, however, the computational requirements are prohibitively large for even functions of very low order.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support of the U.S. Department of Defense High Performance Computing Initiative, the HPC Supercomputer Center at the Army Research Laboratory, and the Modeling & Simulation Research Center of the US Air Force Academy. We would particularly like to thank Steve Senator for his help and support.

A preliminary version of this work by the authors first appeared in abbreviated form in [10].

TABLE 1
CA CLASSIFICATION BY ORDER OF FIRST CORE CONSERVATION FUNCTION

m	CAs with lowest-order, nontrivial conservation functions of order *m*										
1	170	184	204								
2	12	14	15	34	35	42	43	51	140	142	200
3	2	3	4	10	56	76	138	172			
4	1	11	27	29	38	40	72				
5	5	19	24	36	108	132					
6	23	50	77	178	232						
7	-										
8	44	73									
9	7										
10	-										
11	-										
12	33										
13	164										
14	98	104									
15	-										
16	-										

TABLE 2
CONSERVATION FUNCTIONS, BASIS VECTORS AND EQUIVALENCE CLASSES FOR CAS WITH FIRST CORES FROM M=2 THROUGH M=5

	m=2	m=3	m=4	m=5
12	[1 0]		[-1 -1 1 1 0 -2 0 0]	[-5 -5 3 3 1 1 1 1 0 -8 2 2 0 -8 0 0]
14	[1 0]			
15	[1 0]	[-1 1 -2 0]	[1 0 -1 0 -1 0 1 0] [2 2 -5 1 -5 1 0 0]	[-20 4 8 0 4 -4 8 0 4 -4 8 0 12 4 -24 0] [11 0 -7 4 -5 4 -7 0 -3 4 -9 0 -11 0 11 0] [-5 4 1 0 -1 4 -3 0 -3 0 -1 4 1 0 -9 0] [-1 4 -3 0 -1 0 -3 4 -3 4 -1 0 -3 0 -5 0]
34	[1 0]		[-1 -1 1 1 0 -2 0 0]	[-5 -5 3 3 1 1 1 1 0 -8 2 2 0 -8 0 0]
35	[1 0]			
42	[1 0]	[0 0 1 0]	[-1 -1 1 1 1 -1 0 0]	[-3 -3 3 3 0 0 0 0 3 -3 0 0 0 0 0 0] [-2 -2 0 0 2 2 0 0 -1 -3 1 3 0 -6 0 0] [-2 -2 0 0 -1 -1 3 3 -1 -3 4 0 0 -6 0 0]
43	[1 0]			
51	[1 0]	[-1 1 -2 0]	same as f	same as f
140	[1 0]		[-1 -1 1 1 0 -2 0 0]	[-5 -5 3 3 1 1 1 1 0 -8 2 2 0 -8 0 0]
142	[1 0]			
136	[0 1]		[0 0 0 0 0 0 -1 1]	[0 0 0 0 0 0 0 0 -1 -1 -1 3 0 0 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 -3 3]
2		[1 0 0 0]		
3		[1 0 0 0]		[-1 -1 1 1 -2 0 0 0 -2 -2 0 0 0 0 0 0]
4		[1 1 -1 0]		[1 1 1 1 -3 -3 1 1 -1 -1 3 -1 0 0 0 0]
10		[1 0 1 0]	[1 1 0 2 -1 -1 0 0]	
56		[1 1 1 0]		
76		[1 1 1 0]	[1 1 1 1 -2 -2 3 0]	[3 3 3 3 0 0 -12 0 0 0 0 0 0 0 3 0] [0 0 0 0 0 0 0 0 -3 -3 3 3 3 -3 0 0] [0 0 0 0 3 3 -13 3 2 2 0 0 1 3 -3 0]
138		[1 0 1 1]	[-2 -2 0 -3 1 1 1 1]	[0 0 0 0 0 0 1 -1 -1 -1 -1 -1 1 1 1 1]
172		[1 0 0 0]		
1			[1 0 0 0 0 0 0 0]	
11			[1 1 0 1 0 0 0 0]	
27			[1 1 0 1 0 0 0 0]	
29			[1 1 1 1 1 0 0 0]	[-7 -7 -7 -7 3 3 3 3 3 13 -10 -10 -10 0 0 0]
38			[1 1 0 0 0 1 0 0]	
46			[1 1 0 0 0 1 0 0]	
72			[0 0 0 0 1 1 -1 0]	
5				[1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0] [2 2 2 1 1 1 1 1 0 -1 0 -1 0 0 0 0]
19				[0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0]
24				[1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0]
36				[1 1 1 1 -1 0 0 0 -1 -1 0 1 0 0 0 0]
108				[1 1 1 1 1 0 0 1 1 1 0 0 0 1 0 0]
132				[0 0 0 0 1 1 0 0 0 0 -1 0 0 0 0 0]

TABLE 3
EQUIVALENCE CLASSES AND DIMENSIONALITY OF CONSERVATION FUNCTIONS FOR CAS WITH FIRST CORES AT M=2 THROUGH M=6

CA	m=2	m=3	m=4	m=5	m=6	m=7	m=8	m=9	m=10	m=11
12	1		1	1	2	3	5	8	13	21
14	1									
15	1	1	2	4	8	16	32	64	128	256
34	1		1	1	2	3	5	8	13	21
35	1									
42	1	1	1	3	5	9	17	31	57	105
43	1									
51	1	1	2	4	8	16	32	64	128	256
140	1		1	1	2	3	5	8	13	20
142	1									
136	1	1	1	2	4	7	12	21	37	65
2		1			1	1	1	2	3	4
3		1		1	2	3	5	9	16	28
4		1		1	1	2	3	5	8	13
10		1	1		1	3	4	5	9	16
56		1								
76		1	1	3	5	8	14	25	45	82
138		1	1	1	2	4	7	12	21	37
172		1			1	1	1	2	3	3
1			1			1	2	3	4	6
11			1							
27			1			2	3	5	9	16
29			1	1		2	4	7	13	24
38			1			1	2	2	4	8
46			1			1	1	1	2	3
72			1			1	1	1	2	3
5				2	2		2	7	8	21
19				1			1	2	3	4
24				1			1	1	1	2
36				1			1	1	1	2
108				1	1	3	5	7	10	18
132				1		2		4	1	8
23					1		2		4	1
50					1		2		4	1
77					2		4		8	2
178					1		2		4	1
248					2		4		8	2

TABLE 4
EQUIVALENCE CLASSES AND DIMENSIONALITY OF CONSERVATION FUNCTIONS FOR CA'S WITH FIRST CORE AT M=7 THROUGH M=16

CA	m=2	m=3	m=4	m=5	m=6	m=7	m=8	m=9	m=10	m=11	m=12	m=13	m=14	m=15
No CAs with primary cores at m=7														
44							1		1					
73							1							
7								1		2				
No CAs with primary cores at m=10														
No CAs with primary cores at m=11														
33											1			
164												1		
94													1	
104													2	
No CAs with primary cores at m=15														
No CAs with primary cores at m=16														

REFERENCES

[1] Hattori, T., Takesue, S. "Additive conserved quantities in discrete-time lattice dynamical stems," *Physica D,* 49:295-322, 1991.

[2] H. Fuks, "Remarks on the Critical Behavior of Second Order Additive Invariants in Elementary Cellular Automata," Fundamenta Informaticae 78 329-341 (2007), arXiv:nlin.CG/0502037.

[3] Wolfram, S. *A New Kind of Science*, Wolfram Media, Inc., 2002 ISBN 1-57955-008-8.

[4] H. Fuks, "Dynamics of CA rule 142", Complex Systems, 16:123- 138 (2005), arXiv:nlin.CG/0610077

[5] Boykett, T. "Efficient exhaustive listing of reversible one dimensional cellular automata," *Theoretical Computer Science 325(2):* 215-247 (2004).

[6] Boykett, T. "Towards a Noether-like conservation law theorem for one dimensional reversible cellular automata," Elsevier Science preprint.

[7] Pivato, M. "Conservation laws in cellular automata," *Nonlinearity* **15** 1781-1793 (2002).

[8] Morita, K., Imai, K. "Number-Conserving Reversible Cellular Automata and Their Computation-Universality," *Proceedings of the MFCS '98 Satellite Workshop on Cellular Automata,* August 1998, Brno, Czech Republic.

[9] Kotze, L., Steeb, W.H. "Conservation Laws in Cellular Automata," in *Finite Dimensional Integrable Nonlinear Dynamical Systems*, World Scientific Publishing, New Jersey, pp. 333-346 (1998).

[10] Fagin, B., Baird, L. "New Conservation Functions and a Partial Taxonomy for 1-D Cellular Automata", Proceedings of the First IEEE Symposium on Artificial Life, Apr 2007.

NOMENCLATURE

n/a